МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»**
Кафедра математической кибернетики и компьютерных наук

РАЗРАБОТКА КЛИЕНТСКОЙ ЧАСТИ ПРИЛОЖЕНИЯ С ФУНКЦИОНАЛОМ СОЦСЕТИ

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 411 группы
направления 02.03.02 — Фундаментальная информатика и информационные технологии
факультета КНиИТ
Королева Ивана Ивановича

Научный руководитель
зав.каф., доцент, к. ф.-м. н. _____ С. В. Миронов

Заведующий кафедрой
к. ф.-м. н., доцент _____ С. В. Миронов

Саратов 2024

ВВЕДЕНИЕ

В эпоху цифровых технологий социальные сети играют важнейшую роль в жизни современного общества. Они стали не только средством коммуникации, но и платформой для самовыражения, обмена информацией и развития бизнеса. Эффективность и популярность социальной сети во многом зависит от удобства и функциональности её клиентской части, которая определяет взаимодействие пользователя с платформой. В связи с этим, разработка интерфейсов и клиентских приложений, которые бы отвечали современным требованиям и ожиданиям пользователей, является актуальной задачей.

Наша команда разработчиков, участвуя в программе «ВКР как стартап», ставит перед собой цель не просто создания социальной сети, а разработки инновационного веб-приложения, ориентированного на удовлетворение разнообразных потребностей пользователей. Основной задачей является предоставление новых возможностей в области виртуального взаимодействия и обмена контентом. Проект направлен на создание платформы, которая будет способствовать эффективному и безопасному взаимодействию пользователей, интеграции различных сервисов и обеспечению высококачественного пользовательского опыта.

Цель данной дипломной работы — разработка клиентской части приложения с функционалом социальной сети, которое будет соответствовать современным трендам в дизайне и технологиях, а также удовлетворять потребности пользователей в удобной и функциональной платформе для общения и обмена информацией.

Для достижения поставленной цели требуется выполнение следующих задач:

- определение используемых технологий, необходимых для реализации приложения;
- выбор соответствующего инструментария для создания веб-приложения;
- реализация базового функционала клиентской части приложения, включая взаимодействие с сервером и обмен основной информацией для процесса регистрации и авторизации;
- внедрение возможности для пользователей добавлять и удалять контент на портале, включая динамическую подгрузку данных и использование ActionCable;
- реализация механизмов создания и удаления сообщений с использованием

динамического подхода;

— докеризация как клиентской, так и серверной части приложения для обеспечения удобства развертывания и масштабируемости.

Структура и объем работы. Для решения поставленных задач выполнена выпускная квалификационная работа, включающая в себя введение, 2 основные главы, заключение, список использованных источников из 20 наименований и 4 приложения. Работа изложена на 58 страницах.

Первая глава имеет название «Анализ предметной области» и содержит основную информацию о доступных на текущий момент технологий разработки веб-приложений и обеспечения их непрерывной работы.

Вторая глава имеет название «Реализация работы приложения», данная глава содержит подробное описание процесса выполнения работы.

Выпускная квалификационная работа заканчивается заключением, списком использованных источников, а также приложения с кодом А-Г.

Основное содержание работы

Постановка задачи. Основной задачей проекта является создание клиентской части веб-приложения, представляющего собой социальную сеть. Приложение должно предоставлять пользователям возможность создавать и делиться контентом, взаимодействовать через личные сообщения, управлять своим профилем и получать обновления от других пользователей. Ключевыми требованиями к разработке являются высокая производительность, удобство использования и безопасность данных пользователей.

Описание продукта. Разрабатываемый продукт представляет собой социальную сеть, где пользователи могут создавать профили, публиковать посты, комментировать и лайкать публикации других пользователей, а также обмениваться личными сообщениями. Продукт также должен поддерживать различные уровни конфиденциальности, чтобы пользователи могли контролировать, кто может видеть их контент и личные данные.

Анализ существующих решений.

1. Boosty — платформа, предоставляющая возможность творческим людям монетизировать свое творчество через подписки и донаты. Пользователи могут создавать контент, за который подписчики платят деньги, получая взамен эксклюзивный доступ к материалам.
2. Patreon — платформа, которая позволяет творческим людям получать регулярную финансовую поддержку от своих поклонников за доступ к эксклюзивному контенту и преимуществам. Подписчики получают различные уровни доступа в зависимости от размера пожертвований.

Обзор инструментальных средств.

Протокол HTTP. HTTP (HyperText Transfer Protocol) был разработан для передачи документов HTML, но в настоящее время используется для обмена данными любого типа в интернете. В работе с HTTP выделяют три основные категории программ: серверы, клиенты и прокси. Серверы обрабатывают запросы и отправляют ответы, клиенты формируют запросы к серверам, а прокси выступают в качестве промежуточных звеньев, улучшая производительность и безопасность.

Методология MVC. Методология Model-View-Controller (MVC) разделяет приложение на три взаимосвязанных компонента:

1. Model — компонент, отвечающий за данные и бизнес-логику приложения.

Модель управляет данными, правилами и логикой, которые определяют, как данные могут быть изменены или обработаны.

2. View — компонент, который отвечает за отображение данных пользователю. Представление не содержит логики приложения и не изменяет данные напрямую.
3. Controller — компонент, который обрабатывает пользовательский ввод, взаимодействует с моделью и обновляет представление. Контроллеры принимают запросы от пользователей, обрабатывают их и возвращают соответствующие ответы.

Фронтэнд-фреймворки. Svelte и Angular.js являются популярными фреймворками для разработки динамических веб-приложений. Они позволяют создавать интерактивные и высокопроизводительные приложения с богатым пользовательским интерфейсом.

- Svelte — современный фреймворк, который отличается от традиционных подходов к разработке веб-приложений тем, что перемещает большую часть работы по рендерингу на этап сборки, что приводит к более компактному и быстрому коду.
- Angular.js — фреймворк от Google, который предоставляет мощные средства для создания одностраничных приложений с использованием декларативного программирования и инъекции зависимостей.

Выбор инструментов разработки. В результате для выполнения ВКР были выбраны следующие инструменты:

1. HTML — язык разметки, используемый для создания структуры веб-страниц.
2. CSS — язык стилей, используемый для описания внешнего вида и оформления HTML-документов.
3. React.js — библиотека для создания пользовательских интерфейсов, разработанная Facebook. React позволяет создавать компонентный подход к разработке интерфейсов, что упрощает их сопровождение и повторное использование.
4. GraphQL — язык запросов для API, который позволяет клиентам запрашивать только необходимые данные, что снижает объем передаваемой информации и улучшает производительность.
5. Docker — платформа для контейнеризации приложений, обеспечивающая их изоляцию и переносимость. Docker позволяет разработчикам упаковы-

вать приложения и все их зависимости в контейнеры, которые могут быть легко развернуты на любом сервере.

6. Makefile — инструмент для автоматизации сборки проекта. Makefile упрощает выполнение повторяющихся задач, таких как сборка кода, запуск тестов и развертывание приложения.

Реализация работы приложения. Разработка клиентской части веб-приложения социальной сети включает в себя несколько ключевых этапов: установка и настройка окружения для разработки, реализация функциональности авторизации и регистрации, создание ленты новостей, разработка функционала для создания и публикации постов, управление профилем пользователя, обмен личными сообщениями, а также контейнеризация и развертывание приложения.

Первоначальная настройка окружения.

1. **Установка Node.js и npm**. Для начала разработки необходимо установить Node.js и npm (Node Package Manager), которые являются основными инструментами для разработки и управления зависимостями в JavaScript проектах. Эти инструменты позволяют устанавливать библиотеки и фреймворки, необходимые для создания приложения.
2. **Создание структуры проекта**. С помощью утилиты `create-react-app` создается базовая структура проекта. Эта утилита автоматически настраивает все необходимые конфигурации для работы с React, предоставляя готовую основу для разработки.
3. **Настройка Docker**. Для контейнеризации приложения используется Docker. Создаются файлы конфигурации (`Dockerfile` и `docker-compose.yml`), которые описывают окружение и параметры запуска контейнеров. Это позволяет разрабатывать, тестировать и развертывать приложение в изолированной среде, что значительно упрощает управление зависимостями и переносимость кода.

Запуск проекта осуществляется командой `npm run start`. Откроется браузер с основной страницей сайта. Структура проекта будет выглядеть следующим образом:

— `public` — содержит ресурсы, доступные для обработки движком браузера или же поисковыми сервисами. Там находится файл `favicon.ico`, с помощью которого иконка приложения становится видимой на вклад-

ке браузера. Еще он содержит файл `robots.txt`, который отвечает за оптимизацию сайта для поисковых сервисов, а также указывает, какие страницы индексировать.

— `src` — исходный код приложения. Он содержит следующие папки:
  – `assets` — папка, которая содержит основные графические ресурсы, необходимые для стабильной работы сайта, будь то CSS-стили или же изображения, используемые на сайте.
  – `callbacks` — содержит GraphQL-запросы. Внутри нее три папки, разделенные по типам запросов к бэкэнду — mutations (POST-запросы), queries (GET-запросы), и subscriptions (долгосрочные запросы на обновление данных в реальном времени). Каждый файл, содержащийся внутри такой папки, содержит один запрос, который можно вызвать внутри нужного React-компонента.
  – `components` — здесь находятся модули, используемые более одного раза, такие, например, как блок комментариев, хедер (верхняя «шапка» страницы), компонент, ответственный за вывод отдельно взятого поста.
    * `forms` — содержит формы, которые отправляют POST-запрос к бэкэнду, такие как форма входа, форма загрузки фотографий для поста, непосредственно форма создания поста и др.
  – `config` — здесь находятся основные настройки и переменные проекта.
  – `pages` — отвечает за HTML-страницы. Лента новостей, страница профиля и его настроек находятся здесь.

— `App.js` — маршрутизатор: здесь указаны данные в формате «маршрут — React-компонент»:

```
function App() {
  return (
    <BrowserRouter>
      <Routes>
        <Route path="/" element={<LandingPageComponent />} />
  ...
  )
...
}
```

— `index.js` — является «стартовой точкой» любого React-приложения. Вы-

полнение кода клиентского приложения начинается с этого файла.

Процесс авторизации и лента новостей.

1. **Авторизация и регистрация**. Для обеспечения безопасности и контроля доступа пользователей реализованы формы регистрации и авторизации. Пользователи вводят свои данные, которые отправляются на сервер для проверки и сохранения. При успешной авторизации пользователь получает токен, который используется для дальнейшей аутентификации запросов. Вначале пользователь видит страницу входа в аккаунт. Там он может выбрать метод входа — регистрация, вход в аккаунт или авторизация через сторонние сервисы (см. рисунок 1).

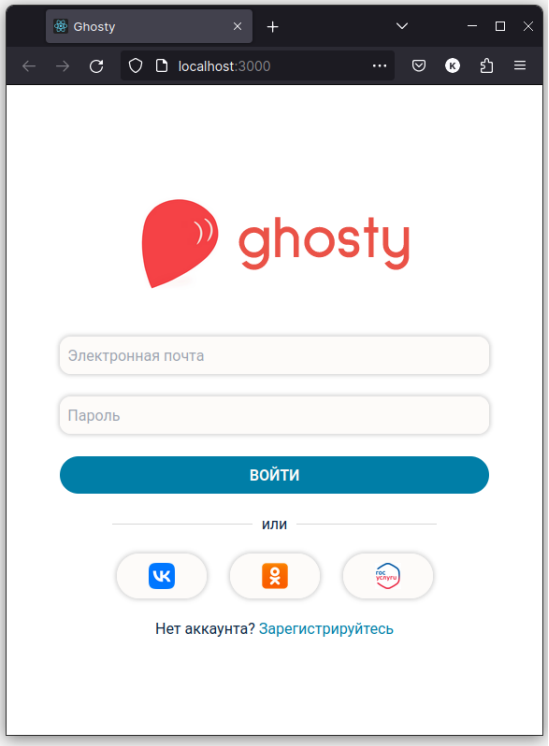

Рисунок 1 – Внешний вид страницы входа в аккаунт

Для успешной работы с ВК необходимо использовать библиотеку vk-openapi, которая позволяет взаимодействовать с API социальной сети и возвращать необходимые данные.

2. **Лента новостей**. Лента новостей представляет собой основной компонент приложения, где пользователи могут видеть обновления от других участников сети. Данные о новых постах загружаются с сервера при загрузке страницы и обновляются в реальном времени, что обеспечивает актуальность отображаемой информации.

Создание новых постов и настройки профиля.

1. **Создание постов**. Создание новых постов вынесено в отдельную форму, аналогичную форме регистрации. Она состоит из двух контроллеров: контроллера для постов и контроллера для загрузки файлов. Загрузка файлов осуществляется через Dropzone. DropZone позволяет превратить в загрузчик файлов любой элемент документа, в том числе и не являющийся частью контейнера form. Её можно настроить на любой вкус, используя CSS и инициализацию объекта. Файловый диалог откроется в том числе и по клику. Если файлы не загрузились по каким либо причинам, то они отмечается крестом, а в подсказке выводится сообщение с расшифровкой ошибки. Скрипт отправляет файлы по AJAX на сервер сразу, не дожидаясь отправки всех данных формы. DropZone обеспечивает только фронтэнд составляющую.

   Пользователи могут создавать новые посты, добавляя текст, изображения и другие типы контента через удобную форму, через которую данные отправляются на сервер, где они сохраняются и становятся доступными другим пользователям в ленте новостей.
2. **Настройки профиля**. В профиле пользователь может изменять личную информацию, такую как имя, электронная почта, биография и настройки конфиденциальности. Это позволяет пользователям персонализировать свои аккаунты и контролировать, кто может видеть их данные и публикации.

   Загрузка изображений через DropZone подразумевает отправку вложенных файлов через Data URL — определённую стандартом схему, которая позволяет включать небольшие элементы данных в строку URL, как если бы они были ссылкой на внешний ресурс. Такой URL формируется на клиенте Далее Data URL разбивается на массив строк с использованием запятой в качестве разделителя. Затем из первой строки массива извлекается информация о MIME-типе изображения. Данные второй строки декодируются из base64 в бинарный формат, затем создается массив типа `Uint8Array`, содержащий байты изображения. Создается объект `File`, который содержит этот массив байтов, имя файла и MIME-тип изображения.

Личные сообщения. Реализация функционала личных сообщений основана на абстракции комнат, сообщений и их участников. Каждое сообщение

и комната ассоциированы с пользовательскими сущностями. Комната способна вмещать множество пользователей посредством учета записей о участниках комнаты и их сообщениях. Сама страница сообщений включает в себя запрос доступных комнат и их вывод через шаблон формы для комнаты.

Для обмена личными сообщениями между пользователями реализован отдельный компонент. Компонент обрабатывает три основных состояния: загрузка данных, наличие ошибки при запросе и успешное получение данных. В случае загрузки данных отображается сообщение о выполнении, в случае ошибки — сообщение с описанием ошибки, а при успешном получении данных происходит итерация по списку чат-комнат. Для каждой чат-комнаты из полученного списка производится рендеринг элемента, содержащего название чат-комнаты и аватар пользователя. Элементы чат-комнаты являются интерактивными и реагируют на нажатие, перенаправляя пользователя на страницу с соответствующим чатом.

Пользователи могут отправлять и получать сообщения в реальном времени, используя WebSocket для мгновенной передачи данных. Это обеспечивает приватное и удобное общение внутри социальной сети.

Контейнеризация и равзертывание приложения.

1. **Докеризация**. Докеризация проекта позволит просто развернуть его на сервере.

   Основную сложность, ввиду использования различных сервисов, составляет упаковка серверной части в контейнер. Рассмотрим ее подробнее.

   Файл `docker-compose.yml` определяет четыре сервиса:

   — `runner`: Сервис для выполнения команд в контейнере. Использует конфигурацию, унаследованную от сервиса backend, и запускает оболочку `/bin/bash`.

   — `rails`: Сервис для запуска веб-приложения на базе Ruby on Rails. Также использует конфигурацию, унаследованную от сервиса backend, и запускает сервер Rails на порту 5000.

   — `postgres`: Сервис для запуска базы данных PostgreSQL версии 16.2 с использованием образа postgres:16.2-alpine3.19.

   — `redis`: Сервис для запуска сервера базы данных Redis версии 6.2 с использованием образа redis:6.2-alpine.

   Сервис `runner` потреуется для того, чтобы была возможность управлять

зависимостями Rails-приложения, а также следить за внутренними процессами внутри контейнера. Для удобства был также написан `Makefile`.
В файле `docker-compose.yml` определены контейнеры, которые будут использованы в дальнейшем. Здесь описан контейнер с сервером Rails, и две базы данных, используемых в проекте (реляционная — PostgreSQL, и нереляционная — Redis). Исходный код этого файла см. в приложении.
После установки откроется терминал внутри созданного контейнера. В нем следует выполнить команду `make install`, чтобы установить все библиотеки для Rails-сервера, а также выполнить стартовые миграции.
Затем следует открыть еще один экземпляр терминала и выполнить там команду `make start`. Это запустит бэкэнд и позволит клиентской части сайта взаимодействовать с ним.
Клиентская часть описывается немного иначе. Аналогично файлу `.gitignore`, для исключения определенных файлов или директорий из процесса сборки в Docker необходим файл `.dockerignore`. Существуют два способа создания этого файла: его можно создать вручную или использовать команду
`touch .dockerignore` в терминале. Файл `.dockerignore` служит для определения списка файлов и директорий, которые Docker должен игнорировать при создании образа контейнера. Этот файл необходим для оптимизации процесса сборки, уменьшения размера образа и исключения из него лишних или чувствительных данных. В нашем случае «лишней» будет папка `node_modules`, которая используется для хранения кеша зависимостей. Она будет создаваться внутри контейнера.
В процессе сборки приложения React для контейнеризации в Docker необходимо выполнить ряд шагов:

а) Импорт Node.js. Первым шагом необходимо импортировать Node.js, поскольку React-приложение требует его для сборки.
б) Определение рабочего каталога. Для удобства работы обычно используется каталог /app.
в) Копирование файлов зависимостей. Для использования команд npm в следующем слое контейнера необходимо скопировать файлы `package.json` и `package-lock.json` в рабочий каталог. Для этого используется команда COPY, после которой указывается точка (.),

означающая текущий каталог.

г) Установка зависимостей. После копирования файлов зависимостей необходимо выполнить установку всех зависимостей, указанных в файле
`package.json`, с помощью команды `npm install`.

д) Копирование остальных файлов. После установки зависимостей следует скопировать остальные файлы проекта в рабочий каталог.

е) Сборка приложения. Для создания сборки приложения выполняется команда `npm build`.

ж) Создание нового образа с использованием Nginx. В этом этапе создается новый образ, который будет использовать Nginx для обслуживания статических файлов.

з) Удаление стандартных статических ресурсов Nginx. Для очистки стандартных статических ресурсов Nginx выполняется соответствующая команда.

и) Копирование статических ресурсов первого этапа. После удаления стандартных ресурсов копируются статические ресурсы из первого этапа сборки приложения.

к) Запуск приложения в контейнере. Наконец, для запуска приложения в контейнере используется инструкция `ENTRYPOINT`, в которой указывается массив строк для запуска приложения.

2. **Развертывание и масштабирование**. Docker Compose используется для управления несколькими контейнерами. С его помощью можно легко запускать, останавливать и масштабировать сервисы приложения. Для развертывания на сервере необходимо лишь несколько команд, что значительно упрощает процесс внедрения новых версий и масштабирование приложения при увеличении нагрузки.

ЗАКЛЮЧЕНИЕ

В ходе данной работы:

- были определены необходимые технологии для реализации приложения;
- выбран соответствующий инструментарий для создания веб-приложения;
- реализован базовый функционал клиентской части приложения, включая взаимодействие с сервером и обмен основной информацией для процесса регистрации и авторизации;
- интегрирована возможность для пользователей добавлять и удалять контент на портале, включая динамическую подгрузку данных и использование ActionCable;
- реализован механизм создания и удаления сообщений с использованием динамического подхода;
- для обеспечения удобства развертывания и масштабируемости были упакованы две основные части приложения — как клиентская, так и серверная.

Таким образом, все поставленные задачи в рамках бакалаврской работы были выполнены.

На российском рынке информационных технологий наблюдается быстрый рост и активная динамика. Этот сектор предоставляет перспективы для занятия устойчивой позиции благодаря развитию мотивации и созданию удобного продукта для пользователей.

Для дальнейшего развития нашего проекта мы планируем привлечь экспертов в области социального медиа-маркетинга (SMM). Этот шаг необходим для успешной реализации нашей бизнес-стратегии.

Наша команда представила свой проект в рамках программы «Стартап как диплом» в Саратовском государственном университете. Это позволило нам получить полезные отзывы от специалистов и расширить наши знания в области стартапов и инновационного предпринимательства.

Благодаря выполнению поставленных задач нам удалось разработать функциональное и безопасное приложение, которое эффективно удовлетворяет потребности пользователей в общении, обмене информацией и поиске интересного контента и знакомств.

В результате получилось полнофункциональное Web-решение, способное работать на множестве устройств. Несмотря на полное выполнение всех постав-

ленных задач, приложение не утратило свою возможность расширения своей структуры: например, дальнейшим шагом в функциональности может стать добавление системы платного контента и управления товарами.